VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Detekce a rozpoznání dopravních značek na křižovatkách

Detection and Recognition of Intersection Road Traffic Signs

Jan Raffai

Bakalářská práce

Vedoucí práce: Ing. Michael Holuša, Ph.D.

Ostrava, 2021

Abstrakt

Cílem této práce bylo rozpoznání svislého dopravního značení na křižovatkách a získání informací z dodatkové tabulky o tvaru křižovatky. Využilo se metody rozpoznání, která je v současnosti považována za nejlepší pro nalezení objektů v obrazech a to detekce pomocí konvolučních neuronových sítí. Samotná implementace je napsána ve frameworku Tensorflow s využitím Object detection API. Následně byl vytvořen soubor dat pro natrénování a otestovaní samotné neuronové sítě. V první části budou popsány základní charakteristiky konvoluční neuronové sítě. Dále popíšeme samotný model užitý pro detekci. V poslední části rozebereme vlastní implementaci a testování v reálných podmínkách.

Klíčová slova

Tensorflow, Object detection API, Python, OpenCV, konvoluční neuronové sítě, Faster R-CNN, Inception, datová sada, dopravní značky, model

Abstract

The goal of the thesis is to create an application, which can detect intersection road traffic signs, and get information out of additional road sign about shape of intersection. To create this detector we will use convolutional neural networks, which is recognized as state-of-the-art method for detecting objects in images. The Implementation itself was written in Tensorflow framework, using Object detection API. Then a dataset for training and testing of the neural network was created. In first part, we will describe basic principles of convolutional neural networks. Next, we will describe model used for detection. In last part, we will go through our implementation and testing in real-time.

Keywords

Tensorflow, Object detection API, Python, OpenCV, convolutional neural networks, Faster R-CNN, Inception, dataset, traffic signs, model

Obsah

Seznam použitých symbolů a zkratek **5**

Seznam obrázků **6**

Seznam tabulek **7**

1 Úvod **8**

2 Neuronové sítě **10**
2.1 Stavba neuronové sítě 10
2.2 Aktivační funkce 11
2.3 Princip učení 13

3 Konvoluční neuronové sítě **14**
3.1 Konvoluční vrstva 14
3.2 Pooling vrstva 15
3.3 Algoritmy pro detekci objektů 15

4 Využité technologie, nástroje a knihovny **20**
4.1 Tensorflow 20
4.2 Tensorflow Object Detection API 20
4.3 LabelImg 21
4.4 OpenCV 22

5 Vytvoření modelu **23**
5.1 Příprava dat 23
5.2 Spuštění trénování 24

6 Testování a zhodnocení výsledků **26**
6.1 Použité druhy značek 26
6.2 Trénovací datová sada 26

6.3 Přesnost modelu 27
6.4 Testování modelu 28
6.5 Rozpoznávání dodatkových tabulí 29
6.6 Zhodnocení výsledků 33

7 Závěr **34**

Literatura **36**

Přílohy **37**

A Obsah archivu **38**

Seznam použitých zkratek a symbolů

ReLu	–	Rectified Linear Unit
CNN	–	Convolutional Neural Network
R-CNN	–	Region-based Convolutional Neural Network
SVM	–	Support Vector Machine
SPP	–	Spatial Pyramid Pooling
RPN	–	Region Proposal Network
COCO	–	Common Objects in Context
mAP	–	Mean Average Precision
VOC	–	Visual Object Classes
IoU	–	Intersection over Union
HSV	–	Hue, Saturation, Value
SSIM	–	Structural Similarity Index

Seznam obrázků

2.1 Schéma základní neuronové sítě 11
2.2 Grafy aktivačních funkcí nelineárního typu - Sigmoid a TanH 12
2.3 Grafy aktivačních funkcí nelineárního typu - ReLu a Leaky ReLu 13

3.1 Konvoluční vrstva [7] 14
3.2 Pooling vrstva [8] 15
3.3 Model R-CNN [9] 16
3.4 Model Fast R-CNN [9] 17
3.5 Model Faster R-CNN [9] 18
3.6 Modul A [10] 18
3.7 Modul B [10] 19
3.8 Modul C [10] 19

4.1 Grafický nástroj LabelImg 21

5.1 Tensorboard TotalLoss s vyhlazením 0,8 pro trénovaný model 25

6.1 Rozpoznávané druhy značek 26
6.2 Ukázka trénovacích dat získaných z kamery v autě 27
6.3 Ukázka přetransformovaného vstupního obrazu 30
6.4 Ukázka nalezení a rozpoznání tvaru křižovatky 30
6.5 Ukázka obsahu datových sad pro dodatkové tabule 31
6.6 Ukázka rozpoznání tvaru křižovatky 31
6.7 Ukázka chybného rozpoznání tvaru kižovatky 32
6.8 Ukázka možného tvaru křižovatky neobsaženého v datových sadách 33

Seznam tabulek

3.1 Porovnání času rozpoznání obecných objektů za jeden snímek 17

4.1 COCO trénované modely, ukázka rozdílů detekce obecných objektů modelů Faster R-CNN [14] 21

6.1 Výsledky testování modelu bez rozpoznávání dodatkových tabulí 28
6.2 Výsledky experimentů změny horní hranice prahu 29
6.3 Výsledky testování modelu s vyhodnocením dodatkových tabulí 32

Kapitola 1

Úvod

Žijeme v moderní době, která mimo rozvoje technologií přináší do lidského života stres, uspěchanost a nervozitu. Z tohoto důvodu se domnívám, že je nezbytné využívat informačních technologií k zdokonalování a kontrole lidské činnosti, jako je bezpečnost v dopravním provozu či předcházení rizika dopravní nehody.

Rozpoznávání dopravních značek je velmi zajímavou oblastí v počítačovém vidění. Důvodem je možnost využití těchto technologií v automobilovém průmyslu, například pro informování řidiče o dopravní značce či pouhé zobrazení užitečných informací o podmínkách na silnici.

Ohromující úspěch technologií hlubokého učení, konvolučních neuronových sítí jako takových, vedlo k posunutí technologického vývoje pro nové aplikace, které by před deseti lety mohly být považovány za nereálné. Plně automatizované řídící systémy jsou intenzivně vyvíjeny v automobilovém průmyslu. Zatímco průmysl se snaží zdokonalit tyto systémy od řídících asistentů, tedy pomoc lidskému řidiči, až po vyšší stupně, kde člověk jako takový je dočasně nebo celkově nahraditelný.

Pro řešení problematiky detekce a klasifikace značek existuje mnoho řešení, jež využívají na tuto problematiku nejrůznější algoritmy. Většina těchto prací má podobné cíle, tedy tvorbu detektoru a klasifikátoru dopravního značení s velmi vysokou přesností a rozpoznáváním v reálném čase. Příkladem takového řešení může být článek publikovaný autory Chin-Chen Chang, Shu-Chun Huang a Huei-Yung Lin v roce 2018 [1], zaměřující se na detekci a klasifikaci dopravního značení pro asistenční systémy při řízení. Pro detekci jsou využívány barevné filtry a selektivní vyhledávání. Je detekován velký počet navržených oblastí, jež jsou zasílány do konvoluční neuronové sítě pro klasifikaci. Dalším příkladem řešení problematiky rozpoznání a klasifikace dopravního značení je metoda publikovaná autory Ali Behloul a Yassmina Saadna v roce 2017, který detekční metody dělí do tří skupin - vyhledávání na základě barev, na základě tvaru či na základě předem natrénovaného modelu. Vyhledávané objekty jsou dále rozděleno do více kategorií pro usnadnění učení, detekce i klasifikace dat.

V této práci bude popsán stručný úvod do neuronových sítí jako takových, jejich speciální variantě v podobě konvolučních neuronových sítí, jež byly využity při tvorbě detektoru a typy algoritmu

pro detekci objektů. Kromě toho budou také popsány knihovny a nástroje, které byly využity. V neposlední řadě bude popsána vlastní implementace detektoru. Závěrem bude popsáno testování, bude řečeno jaké značky jsou rozpoznávány, analyzovány chyby spojené s klasifikací objektů, analyzováno rozpoznávání dodatkových tabulí a v neposlední řadě zhodnoceny výsledky práce.

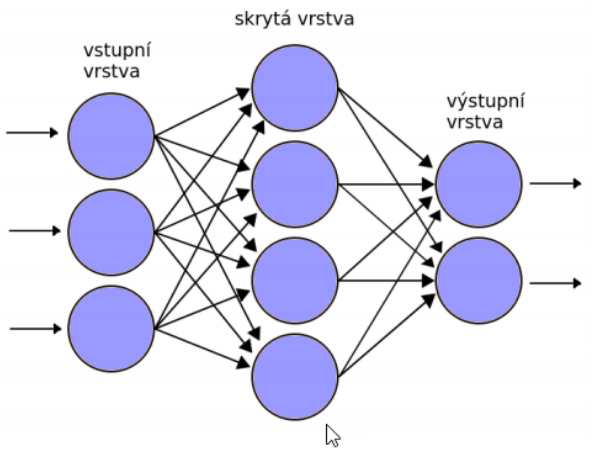
Kapitola 2

Neuronové sítě

V dřívějších dobách bylo zjištěno, že mozek je tvořen velkým množstvím vzájemně propletených buněk, které nazýváme neurony. Tyto buňky spolu komunikují pomocí elektrických impulzů. Od vzniku prvních počítačů se dlouhou dobu snažili programátoři vytvořit počítačový program, který by byl co nejvíce podobný našemu mozku v tom smyslu, že by uměl myslet a mohl se učit z vlastních chyb. Tyto programy spadají do skupiny umělé inteligence. Nejlepším kandidátem na vývoj umělé inteligence jsou prozatím neuronové sítě, berou si inspiraci v přírodě a hlavně v lidském mozku, jehož nejdůležitější vlastností je schopnost učení [2].

2.1 Stavba neuronové sítě

Základní stavební jednotkou neuronových sítí je umělý neuron, resp. jeho matematický model. Umělé neuronové sítě jsou následně tvořeny souborem modelů neuronů, určitým způsobem vazaných a mezi sebou vzájemně propojených. V praxi je nejčastěji používaným modelem McCulloch-Pittsův neuron, označován jako základní model neuronu. Umělý neuron byl inspirován přírodním neuronem, ve smyslu, že obsahuje konečný počet vstupů xn (*př. x1, x2*) a pouze jeden výsledný aktivační signál stejně jako biologický neuron [3]. Funkce biologického neuronu jsou v umělém neuronu nahrazeny matematickými modely, jichž existuje velké množství, lišící se svou topologií a typy matematických funkcí, které popisují jejich chování. V každém neuronu se vstupní hodnoty transformují na výstup pomocí výpočtu vstupního potenciálu y a aktivační funkcí f.



Obrázek 2.1: Schéma základní neuronové sítě

2.2 Aktivační funkce

Aktivační funkce je důležitou součástí neuronu, která určí, zda byl neuron excitován, neboli aktivován. Excitovaný neuron posílá výsledek výpočtu neuronům, které jsou na něj napojeny. Funkce mapují výsledky výpočtů mezi hodnotami 0 až 1 nebo -1 až 1, dle průběhu aktivační funkce je interval otevřený nebo uzavřený. Bývají rozděleny do tří skupin a volí se na základě užití. První skupinu tvoří ostrá nelinearita (Binary Step), druhou zastupují aktivační funkce lineární a třetí funkce nelineární. Uvedeme si základní a velmi často využívané funkce.

2.2.1 Ostrá nelinearita

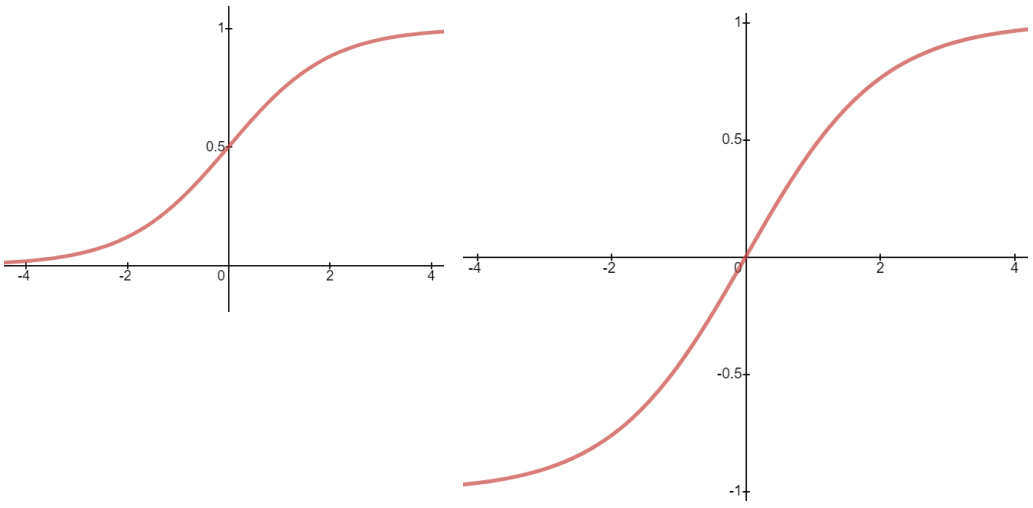
Ostrá nelinearita je funkce, jež se aktivuje změnou prahu. Pokud se vstupní hodnota vyskytuje nad nebo pod určitou prahovou hodnotou, je neuron aktivován a posílá stejný signál další vrstvě. Problémem nicméně zůstává, že ostrá nelinearita nepovoluje více hodnotové výstupy, nepodporuje rozdělení vstupů do více skupin. Hodí se pouze pro rozdělení případů do dvou kategorií [4].

2.2.2 Standartní sigmoida (Sigmoid)

$$f(x) \quad = \quad \frac{1}{1+e^{-x}}$$

Sigmoida nabízí narozdíl od ostré nelinearity nekonečný počet výstupů, tedy lze rozdělit vstup do více skupin. Výstupní hodnoty se pohybují mezi 0 a 1. Zvlášť je využívána pro modely, kde je zapotřebí na výstupu předpovědět pravděpodobnost. Jelikož pravděpodobnost obecně existuje v rozmezí 0 až 1, je sigmoida tou správnou volbou pro klasifikaci více než dvou tříd [5]. Pro velmi

vysoké nebo velmi nízké hodnoty x není téměř žádná změna pro předpovědi, což může mít za následek odmítnutí sítě dalšího učení [4].



(a) Graf aktivační funkce - sigmoida (b) Graf aktivační funkce - hyperbolický tangens

Obrázek 2.2: Grafy aktivačních funkcí nelineárního typu - Sigmoid a TanH

2.2.3 Hyperbolický tangens (TanH)

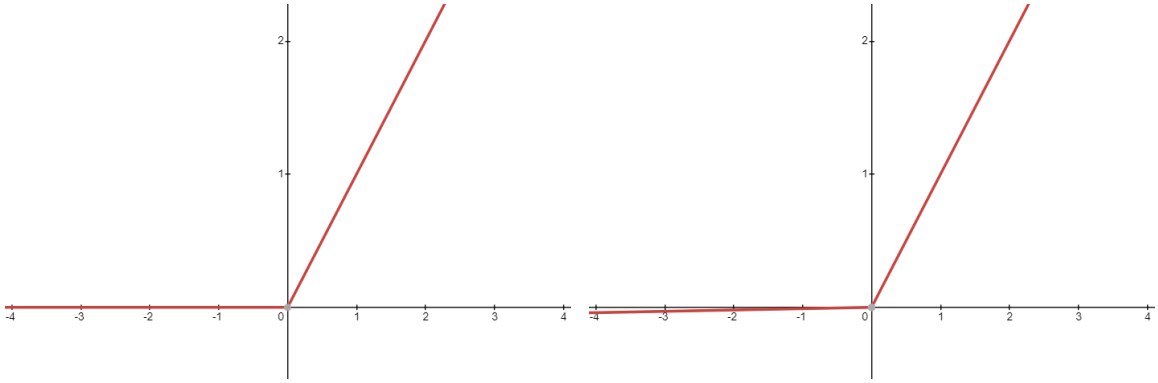
$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Hyperbolický tangens se velmi podobá sigmoidě, nicméně pravděpodobnost existuje v rozmezí od -1 do 1, což je výhodné, chceme-li zaznamenávat hodnoty menší než 0. Nulové vstupy jsou mapovány v blízkosti 0 v grafu. Má stejný problém se zpomalením učení jako funkce Sigmoid [4].

2.2.4 ReLu

$$f(x) = \begin{cases} x & pokud \quad x > 0 \\ 0 & pokud \quad x \leq 0 \end{cases}$$

ReLu (Rectified Linear Unit) je v současnosti nejvyužívanější aktivační funkcí, neboť je využívána v téměř všech konvolučních neuronových sítích nebo algoritmech hlubokého učení. Její výstup je v rozmezí 0 až ∞. Neregistruje žádné chyby spojené se zpětným průchodem (tzv. backpropagation) oproti výše zmíněným. Problémem je okamžité nastavení záporných hodnot na 0, což razantně snižuje schopnost modelu správně trénovat z předvytvořených dat [6]. Pokud by v průběhu trénování měl neuron zápornou hodnotu, kvůli nastavení výstupní hodnoty na 0 se není schopen z tohoto stavu obnovit [4].



(a) Graf aktivační funkce - ReLu (b) Graf aktivační funkce - Leaky ReLu

Obrázek 2.3: Grafy aktivačních funkcí nelineárního typu - ReLu a Leaky ReLu

2.2.5 Leaky ReLu

$$f(x) = \begin{cases} x & pokud \quad x > 0 \\ 0,01x & pokud \quad x \leq 0 \end{cases}$$

Jedná se o vyřešení problému funkce ReLu její úpravou, místo nastavení vstupní hodnoty na 0 pro všechna záporná čísla dovoluje malé záporné hodnoty. Díky tomu je neuron po „odumření“ schopen se zotavit, jelikož jeho hodnota není pevně změněná na 0 [6].

2.3 Princip učení

Učení obstarává adaptační algoritmus, který nastavuje váhy. Nastavení vah odpovídá nalezení nejvýstižnější transformace vstupních vektorů na výstup. Váhy jsou získány opakovanými průchody sítí. Velkou výhodou neuronové sítě je schopnost nalezení správné transformace i v případech, kdy je to analyticky jen stěží řešitelné nebo přímo neřešitelné. Postačuje k tomu velké množství tréninkových dat, na kterých se zvládne daná síť správně naučit.

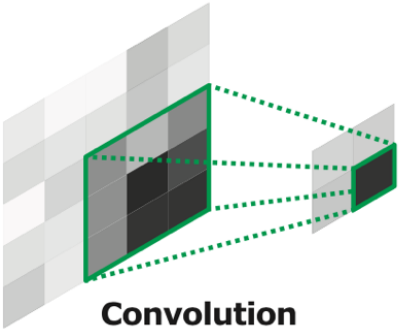
Kapitola 3

Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou podmnožinou neuronových sítí. Mají veškeré vlastnosti popsány dříve, liší se nicméně ve vstupní vrstvě. Vyznačují se použitím konvolučních a pooling vrstev, díky nimž jsou schopny efektivně zpracovávat vstupy velkých rozměrů, jako jsou například obrázky nebo zvuky. Tento typ sítí je využíván pro detekci a následnou klasifikaci objektu z obrazu.

3.1 Konvoluční vrstva

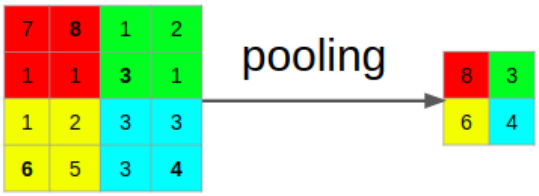
Konvoluční vrstva obsahuje více konvolučních jader, tato jádra se liší ve svých hodnotách, nicméně mají stejnou velikost, nejčastěji o rozměrech 3×3 buněk. Při zpětném průchodu konvoluční vrstvou jsou upravovány převážně hodnoty tvořící jádra. Na počátku procesu učení bude docházet k velkým změnám v jádrech, neuronová síť se bude snažit najít jádro, které bude nejlépe detekovat určitý příznak typický pro detekovaný objekt.



Obrázek 3.1: Konvoluční vrstva [7]

3.2 Pooling vrstva

Funkcí pooling vrstvy je postupné zmenšování velikosti vstupů, snížení počtu parametrů a výpočtů v neuronové síti [8]. Pracuje na každé aktivační mapě, tvaru matice, samostatně a mění její prostorovou velikost užitím operace pro získání maximální hodnoty matice, viz. obrázek 3.2. Nejvyšší hodnota je zapsána do nově vzniklé výstupní matice. Abychom mohli z nové matice získat matici o původní velikosti, je nutné při zmenšení zaznamenat pozici maximalní hodnoty.



Obrázek 3.2: Pooling vrstva [8]

3.3 Algoritmy pro detekci objektů

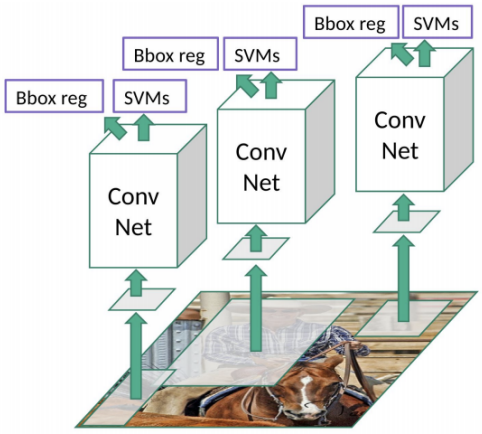
V této kapitole budou představeny algoritmy konvoluční neuronové sítě, které oproti klasickým konvolučním neuronovým sítím, jež jsou schopny pouze rozpoznat objekt v obraze, tyto algoritmy navíc umožňují i lokalizaci objektů. R-CNN jsou v současnosti velmi populární typy detekčních algoritmů, převážně kvůli jejich přesnosti a samotné rychlosti detekce, díky čemuž jsou vhodným detekčním algoritmem pro rozpoznávání dopravního značení. Výstupem je sada označených oblastí, ve kterých byly detekovány objekty, v jednom obraze může být více takto detekovaných objektů.

3.3.1 Region-based Convolutional Neural Network (R-CNN)

Problémem klasické konvoluční neuronové sítě je přístup, resp. vyhledávání objektů, jelikož mohou mít jiné pozice a může jich být označených různý počet. K nalezení by bylo zapotřebí procházení velkého množství oblastí, což by mohlo znamenat odrovnání stroje výpočetně. Pro vyřešení problému výběru přilišného množství potenciálních oblastní navrhl Ross Girshick metodu, ve které užíváme vyhledání výběrem pouze 2000 potenciálních oblastí z obrazu [9]. Díky tomu, nyní místo procházení obrovského množství oblastí postačí pracovat s oblastmi o určitém počtu. R-CNN nejprve začne procházet obraz pro možné kandidáty pomocí algoritmu selektivního vyhledávání, jenž vygeneruje okolo 2000 potenciálních oblastí. Tyto oblasti jsou vloženy do konvoluční neuronové sítě, jejíž výstupem je 4096 rozměrný znakový vektor. CNN získá vlastnosti obrazu a vloží je do SVM pro klasifikaci přítomnosti objektu v potenciální oblasti, jež je vidět na obrázku 3.3.

Velkou nevýhodou R-CNN je masivní doba potřebná pro natrénování sítě, kvůli procházení 2000 potenciálních oblastí na jeden obrázek. Nemůže být ani implementována pro detekci v reálném

čase, neboť snímek o velikosti 600 × 600 pixelů trvá modelu rozpoznat kolem 47 vteřin. Algoritmus selektivního vyhledávání navíc nepodporuje učení.



Obrázek 3.3: Model R-CNN [9]

3.3.2 Fast R-CNN

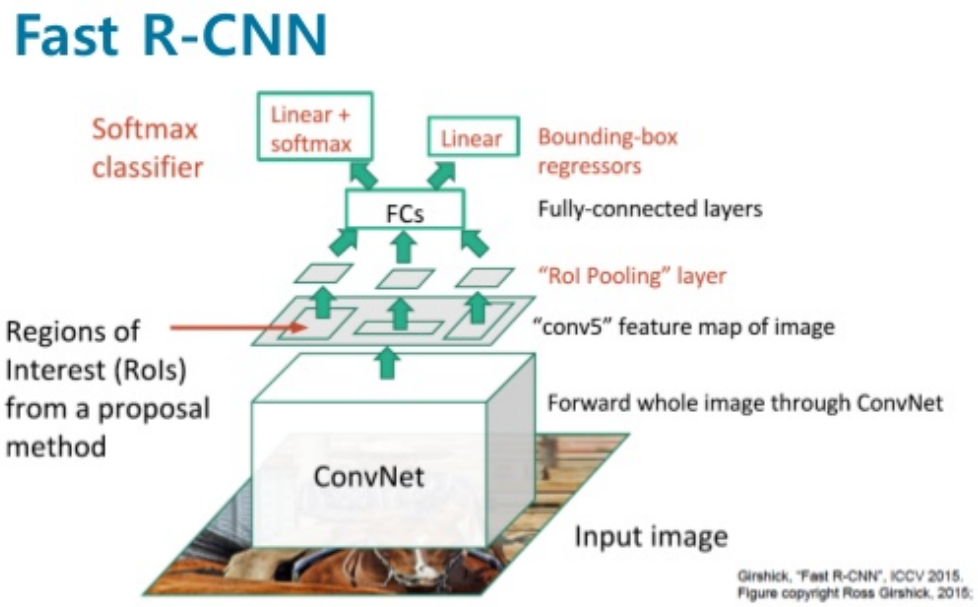
Algoritmus Fast R-CNN je v mnoha ohledech podobný R-CNN, jelikož autorem je stejná osoba. Přístup algoritmu se velmi podobá R-CNN, jenže místo vkládání potenciálních oblastí do CNN, je využíván vstupní obrázek, ze kterého se generuje konvoluční aktivační mapa. [9] Z této mapy jsme schopni určit oblasti, úpravou je vložit do čtverců a za použití RoI pooling layer přetvořit vstupy o libovolné velikosti na výstup s pevnou délkou z důvodu omezení velikosti v plně propojených vrstvách. Softmax vrstva, která nahradila předchozí SVM, je použita pro předvídání třídy v potenciální oblasti a pro posun hodnot ohraničujících boxů, viz. obrázek 3.4.

Oproti předchozímu R-CNN modelu se detekce zrychlila 25násobně, což u snímku o velikosti 600 × 600 pixelů odpovídá přibližně 2 sekundám.

Problémem předchozího algoritmu R-CNN bylo obtížné trénování sítě, kvůli 3 různým částem, které v tomto novém provedení jsou spojeny do jedné architektury.

3.3.3 Faster R-CNN

Předchozí zmíněné algoritmy používají selektivní vyhledávání pro nalezení potenciálních oblastí. Tento proces je velmi pomalý a náročný na čas, což ovlivňuje výkon neuronové sítě. Proto Shaoqing Ren přišel s algoritmem pro detekci objektů, který odstraní selektivní vyhledávání a místo toho nechá síť naučit návrhy oblastí. [9] Z obrazu daného na vstupu se vytváří konvoluční aktivační mapa, podobně jako tomu bylo u Fast R-CNN. Oddělená síť nyní předpovídá potenciální oblasti, což je rozdíl oproti předchozímu algoritmu, kde byl využit algoritmus selektivního vyhledávání pro



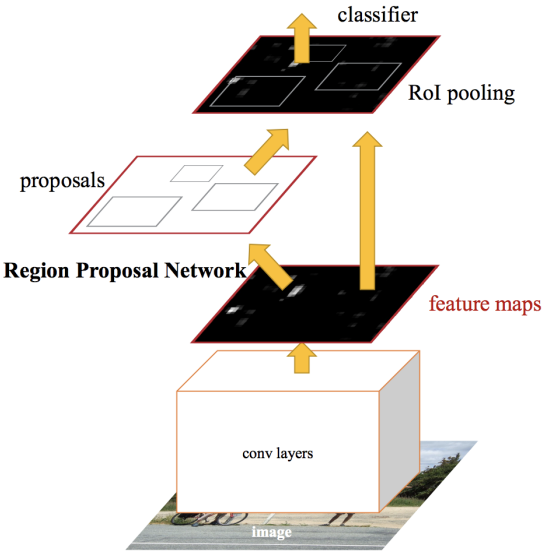
Obrázek 3.4: Model Fast R-CNN [9]

identifikaci těchto oblastí. Tato nově přidaná síť se jmenuje RPN (Region Proposal Network), viz. obrázek 3.5. Počet potencionálních oblastí se také díky nové síti snížil z 2000 na 300.

Původně byl algoritmus příliš náročný na výpočty a zároveň detekce jednoho snímku trvala 47 vteřin. S každým algoritmem bylo přineseno vylepšení, které zmenšilo náročnost na paměť, urychlilo detekci a klasifikaci objektů ze snímků, zjednodušilo trénování sítě samotné a v neposlední řadě zmenšilo počet procházených oblastí. Porovnání rychlosti rozpoznání značek je vidět v tabulce 3.1. Rychlosti jsou uváděny pro obrázky o velikosti 600 × 600 pixelů, trénované na grafické kartě Nvidia GeForce GTX TITAN X.

Tabulka 3.1: Porovnání času rozpoznání obecných objektů za jeden snímek

Algoritmus	Čas potřebný pro rozpoznání (s)
R-CNN	47
Fast R-CNN	2,1
Faster R-CNN	0,2



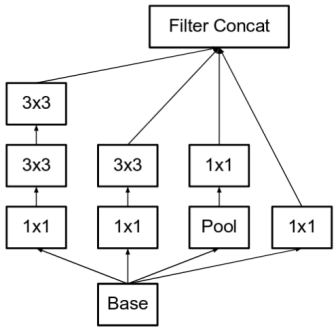
Obrázek 3.5: Model Faster R-CNN [9]

3.3.4 Inception V2

Většina populárních konvolučních sítí skládala konvoluční vrstvy co nejvíce do hloubky s cílem získat lepší výsledky. Inception na druhou stranu je komplexnější. Užívá mnoho triků za účelem posunout výkon neuronové sítě rychlostí i přesností.

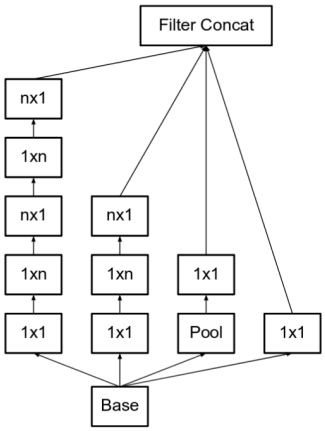
Předpokladem Inception V2 bylo, že neuronové sítě fungují lépe, když se rozměry vstupu nemění příliš drasticky. Přílišné zmenšení by mohlo vést ke ztrátě informací.

Inception V2 je modul navržený ke snížení složitosti konvoluční sítě. Má 3 rozdílné typy modulů, které nemají žádné stanovené pojmenování. Pro nás to tedy budou moduly A, B a C. První z nich (modul A, viz. obrázek 3.6) nahradil konvoluci 5×5 na dvě konvoluční operace o velikosti 3×3 buňky. I když se to může zdát nepochopitelné, konvoluce 5×5 buněk je téměř třikrát náročnější než konvoluce 3×3 buňky, proto složením dvou těchto konvolucí dochází ke zvýšení výkonu.



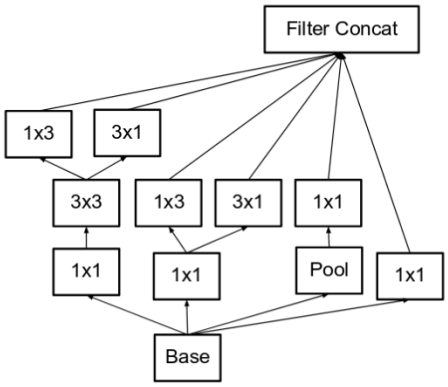
Obrázek 3.6: Modul A [10]

Druhý modul na obrázku 3.7, pro nás pojmenovaný modul B, nahradil konvoluce o velikosti n × n kombinací 1 × n a n × 1. Příkladem může být konvoluce o velikosti 3 × 3 buňky, jež je stejná jako 2 konvoluční operace, z nichž jedna je o velikosti 1 × 3 a druhá o velikosti 3 × 1 buňky. Bylo zjištěno, že touto metodu je ušetřeno 33% výpočetního výkonu.



Obrázek 3.7: Modul B [10]

Ztráta informací byla v modulu C minimalizována užitím širších skupin filtrů místo skupin filtrů do hloubky. Pokud by totiž modul byl vytvořen do hloubky, nastalo by přílišné zmenšení rozměrů a tím pádem i ztráta dat. Modul je zobrazen na obrázku 3.8.



Obrázek 3.8: Modul C [10]

Kapitola 4

Využité technologie, nástroje a knihovny

V této kapitole budou popsány technologie, grafické nástroje a některé knihovny využité při vytváření detektoru. Nejprve bude popsána knihovna Tensorflow, která byla využita pro natrénování modelu. Následovně popíšeme grafický nástroj LabelImg, jež posloužil pro označení tříd pomocí boxů v trénovací datové sadě. V neposlední řadě bude popsána OpenCV knihovna použitá při detekci a klasifikaci dat ve skriptech. Pro vytvoření modelu byla použita knihovna Tensorflow verze 1.5 a CUDA knihovny - CUDA Toolkit v8.0 a cuDNN v7.0.5.

4.1 Tensorflow

Tensorflow je open source softwarová knihovna pro vysoce výkonné numerické výpočty. Jedná se o druhý framework strojového učení společnosti Google, který byl původně vyvinut výzkumníky a inženýry z Google Brain týmu, vytvořený a využitý pro návrh, stavbu a trénování modelů hlubokého učení. Knihovny Tensorflow jsou schopny udělat numerické výpočty s grafy toku dat [11]. Flexibilní architektura systému umožňuje snadné nasazení výpočtů a vývoj aplikací na různých platformách. Podporuje vývoj v jazycích Python a C++, pro ostatní programovací jazyky není zpětná kompatibilita garantována [12].

4.2 Tensorflow Object Detection API

Vytvoření modelů strojového učení schopných lokalizace a identifikování více objektů z jediného obrazu zůstává i nadále hlavním úkolem počítačového vidění. TensorFlow Object Detection API je open source framework postavený na TensorFlow, který zjednodušuje konstrukci, trénování a nasazení modelu pro detekci objektů [13].

Tabulka 4.1: COCO trénované modely, ukázka rozdílů detekce obecných objektů modelů Faster R-CNN [14]

Jméno modelu	Rychlost (ms)	COCO mAP	Výstup
faster rcnn inception v2 coco	58	28	Boxy
faster rcnn resnet50 coco	89	30	Boxy
faster rcnn resnet50 lowproposals coco	64		Boxy
faster rcnn resnet101 coco	106	32	Boxy
faster rcnn resnet101 lowproposals coco	82		Boxy
faster rcnn inception resnet v2 atrous coco	620	37	Boxy
faster rcnn inception resnet v2 atrous lowpr. coco	241		Boxy
faster rcnn nas	1833	43	Boxy
faster rcnn nas lowproposals	540		Boxy

Rychlost v tabulce 4.1 je uváděná v milisekundách pro obraz o velikosti 600 × 600 pixelů trénované na grafické kartě Nvidia GeForce GTX TITAN X, mAP (`Mean Average Precision`) je hodnota přesnosti specifická pro COCO set, čím je vyšší tím přesnější je model. Výstupem mohou být boxy nebo masky [14].

4.3 LabelImg

LabelImg je grafický nástroj, který je určen pro anotaci obrazu. Veškeré obrázky, které chceme anotovat, jsou načteny a následně jim je možné vybrat adresář pro uložení výstupního souboru. Tento adresář je přednastaven na adresář vstupní. Tlačítkem Create RectBox je vybrána oblast v niž se objekt nachází a je ji přiřazena příslušná třída. Anotace jsou ukládány do XML souborů ve formátu PASCAL VOC, viz. výpis 4.1, vázající se na příslušný obrázek. Podporuje také formát You Only Look Once (YOLO) [15].

Obrázek 4.1: Grafický nástroj LabelImg

4.4 OpenCV

OpenCV je multiplatformní knihovna zaměřená především na počítačové vidění a zpracování obrazu v reálném čase [16]. Byla vyvinuta za účelem poskytnutí běžné infrastruktury aplikacím s počítačovým viděním a pro rozšíření strojového učení mezi komerčními produkty.

Knihovna obsahuje přes 2500 optimalizovaných algoritmů, zahrnující komplexní sadu pro počítačové vidění a strojové učení. Algoritmy mohou být využity pro detekci a rozpoznání tváří, k identifikaci objektů, klasifikování lidských pohybů ve videích, sledování pohybu kamery a objektů, extrakci 3D modelů z objektů a jiné. Existují rozhraní pro C++, Python, Java a MATLAB a je podporováná operačními systémy Windows, Linux, Mac OS a Android.

```
<annotation>
   ...
   <filename>priority1.jpg</filename>
   ...
   <size> ... </size>
   <segmented>0</segmented>
   <object>
      <name>prioritySign</name>
      <pose>Unspecified</pose>
      <truncated>0</truncated>
      <difficult>0</difficult>
      <bndbox>
         <xmin>510</xmin>
         <ymin>212</ymin>
         <xmax>547</xmax>
         <ymax>241</ymax>
      </bndbox>
   </object>
</annotation>
```

Listing 4.1: Jeden z vstupů programu LabelImg v PASCAL VOC formátu

Kapitola 5

Vytvoření modelu

V této kapitole bude vysvětlen postup při přípravě dat a následném natrénování COCO modelu na těchto datech založeném. Pro detekci byl zvolen algoritmus konvoluční neuronové sítě, `Faster R-CNN Inception v2`. Pokud bychom porovnali COCO modely v tabulce 4.1, zjistili bychom, že Inception druhé verze má nejlepší poměr přesnosti detekce za určitý čas. Z tohoto důvodu bylo tomuto modelu dána přednost před `resnet50`. Model byl natrénován na počítači s operačním systémem Linux.

5.1 Příprava dat

Pro spuštění trénování je nejprve zapotřebí vytvoření robustní datové sady obrázků rozdělených do dvou skupin. První skupinu tvoří obrázky určeny pro trénování modelu, ve druhé jsou obrázky pro testování. Tyto skupiny by měly mít rozděleny data v poměru 80% pro trénování ku 20% pro testy. V mém případě byl vytvořen nový adresář images s podadresáři train a test. Po vytvoření a rozdělení do příslušných adresářů, musí být datové sady označkovány. Pro jejich označení využijeme grafický nástroj LabelImg popsaný v předchozí kapitole. Při každém uložení změn provedených v nástroji, je vytvořen XML soubor ve formátu PASCAL VOC příslušící danému obrazu. Data z příslušných XML souborů byla dle datové sady uložena do dvou CSV souborů. S již označenými obrázky je na čase převést tato data do doporučeného formátu TFRecord, jež slouží jako vstupní data pro trénovací model TensorFlow. V tomto formátu jsou data uložena binárně a optimalizována pro použití s Tensorflow.

Pro rozdělení tříd při trénování a testování modelu je potřeba vytvořit značkovací mapu s příponou .pbtxt, kde jsou namapovány všechny třídy a k nim odpovídající identifikační čísla. Část mapy je zobrazena ve výpise 5.1.

```
item {
  id: 1
  name: 'prioritySign'
  display_name: 'Hlavní cesta'
}

item {
  id: 2
  name: 'yieldSign'
  display_name: 'Dej prednost v jizde'
}

item {
  id: 3
  name: 'stopSign'
  display_name: 'Stuj, dej prednost v jizde'
}
...
```

Listing 5.1: Ukázka značkovací mapy s příponou .pbtxt

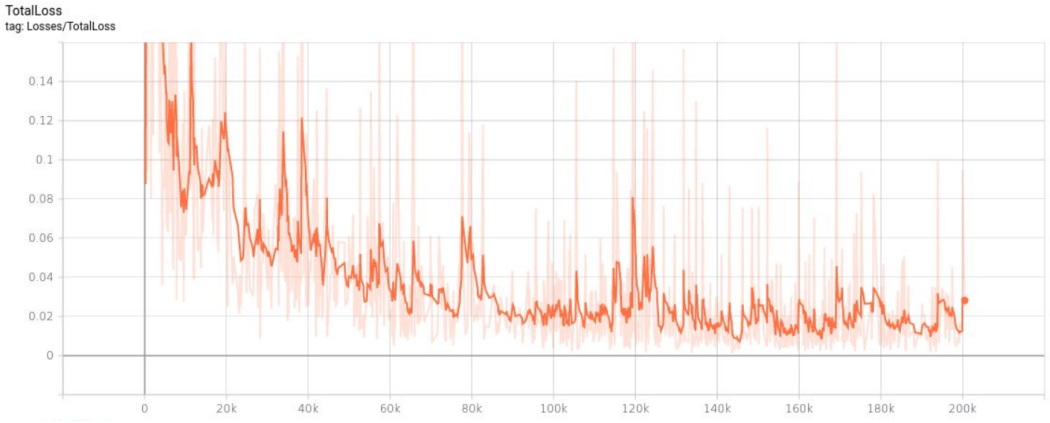
5.2 Spuštění trénování

V průběhu trénování jsou vytvářeny soubory obsahující informace o uložených hodnotách, z nichž ten poslední je na konci trénování využit pro vygenerování inferenčního grafu potřebného pro samotnou detekci.

Každý krok tréninku zaznamenává ztrátu způsobenou detekcí, nejprve je tato hodnota vysoká, postupným tréninkem nicméně klesá. Pro model využitý v této práci, tj. Faster R-CNN Inception v2 byla původně hodnota ztráty kolem 2,5. Je doporučováno trénovat model do té doby, než ztráta všech kroků je permanentně menší než 0,05, což v mém případě trvalo něco okolo 170 tisíc kroků.

Postup trénování je možné sledovat z příkazové řádky, popř. užitím TensorBoard. Tento nástroj zobrazuje vývoj tréninku s pomocí grafů, z nichž nejdůležitějším je graf ztráty dat ukazující celkovou ztrátu klasifikátoru v čase, viz obrázek 5.1

S průběhem trénování jsem měl v tomto případě problém, jelikož se mi nepodařilo využít Tensorflow-GPU pro natrénování. Díky tomu byla rychlost kroku mnohonásobně menší než mohla být. Předpokládaná doba na tomto stroji byla v průměru 0,65s na krok, kdežto mě jeden krok vyšel



Obrázek 5.1: Tensorboard TotalLoss s vyhlazením 0,8 pro trénovaný model

přibližně na 3s. Trénování čistě na CPU nemá žádný vliv na hodnotu ztráty, nicméně razantně sníží rychlost tréninku.

Kapitola 6

Testování a zhodnocení výsledků

V této kapitole bude popsáno testování modelu, zhodnocena přesnost získána z testovaných dat. Následně bude popsáno rozpoznání dodatkových tabulek a nakonec zhodnoceny výsledky.

6.1 Použité druhy značek

V práci byly detekovány čtyři druhy značek rozdělených do dvou skupin. První kategorii představovaly značení hlavní cesty, tedy značení hlavní pozemní komunikace na obrázku 6.1a a křižovatka s vedlejší pozemní komunikací, jenž se nachází mimo města. Zobrazená je na obrázku 6.1b. Druhou skupinu pak tvoří značení vedlejších cest, dej přednost v jízdě 6.1c a stůj, dej přednost v jízdě 6.1d. Každá z těchto značek může být dále rozšířena o dodatkovou tabulku s vyznačením tvaru křižovatky.

(a) Hlavní pozemní komunikace (b) Křižovatka s vedlejší pozemní komunikací (c) Dej přednost v jízdě (d) Stůj, dej přednost v jízdě

Obrázek 6.1: Rozpoznávané druhy značek

6.2 Trénovací datová sada

Před vytvářením detektoru objektů z obrazu, bylo potřeba získat data, která budou použita pro trénování a testování. Pro vytvoření robustního klasifikátoru je zapotřebí velké množství dat, která se budou lišit. Pro správné natrénování modelu by data obsažená v trénovací datové sadě měla obsaho-

vat rozdílná pozadí, světelné podmínky, náhodné objekty, či rozdílná roční období. Kvalita této sady může razantně ovlivnit kvalitu našeho výsledného modelu. Pokud bude datová sada nevyvážená, je větší pravděpodobnost, že klasifikátor určí chybně stav značky.

Trénovací datová sada pro trénování detektoru je složena z 2153 jpg snímků, viz. obrázek 6.2, které byly shromážděny pomocí fotek pořízených v reálném životě, z Google Street View a snímků získaných z videí.

Obrázek 6.2: Ukázka trénovacích dat získaných z kamery v autě

Pro získání dat z datové sady byl použit grafický nástroj `LabelImg` dříve popsaný v této práci.

6.3 Přesnost modelu

Pro porovnání přesnosti byly použity jednotky `Precision`, `Recall` a `F1 Score`, viz. tabulka 6.1. Precision udává hodnotu správnosti modelu při předpovídání pozitivní třídy. Recall je jednotka pro zjištění, kolik z pozitivních tříd bylo správně určeno a F1 skóre je přesnost testu. Pro výpočet přesnosti modelu je potřeba rozdělit výsledky detekce na 4 stavy:

- Pravdivě pozitivní (TP) = správně určená třída, tzn. bylo vyhodnoceno, že značka je hlavní cesta a značka jí skutečně je
- Pravdivě negativní (TN) = ve vyhledávané oblasti nebyla nalezena požadovaná třída, tzn. bylo vyhodnoceno, že značka není hlavní cestou a značka jí skutečně není
- Falešně pozitivní (FP) = chybně určená třída, tzn. značka byla detekována v místě, kde se ve skutečnosti nenachází

- Falešně negativní (FN) = nenalezení třídy v oblasti, ve které se vyskutuje, tzn. značka nebyla v obraze nalezena, i když se zde nachází

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F_1 Score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

Tabulka 6.1: Výsledky testování modelu bez rozpoznávání dodatkových tabulí

Název jednotky	Hodnota
Pravdivě pozitivní	88,7%
Falešně pozitivní	2,9%
Falešně negativní	8,4%
Precision	0,968
Recall	0,914
F1 Score	0,940

6.4 Testování modelu

Pro detekci byl použit a následně natrénován model Faster R-CNN Inception v2 na 200 tisíc kroků, což z důvodů potíží trvalo mnohem déle než bylo předpokládáno. Ze mnou nejasných důvodů byl model nejspíš trénován pouze na CPU, což by dokonale vysvětlilo 7násobně delší čas trénování. Detektor byl trénován přibližně 5 dnů a 13 hodin. Trénování probíhalo na školním serveru merlin1, který ač disponuje dvěma grafickými kartami NVIDIA Ge-Force RTX 2080, nebyl model pravděpodobně trénován ani na jedné z nich. Celkové ztráty klesly na 200 tisících krocích pod 0,03, viz. obrázek 5.1.

Pro výsledné otestování a zjištění přesnosti rozpoznávání bylo užito 464 obrázků z nezávislé datové sady a jedno sestříhané video. Nad každým jednotlivým obrazem byla testována detekční síť, která vyobrazila ohraničující boxy v místech, kde skóre klasifikace bylo vyšší než 85%. Kvůli nedostatečně velké datové sadě pro stopku, nejsou tyto značky vždy vyhodnoceny s dostačující přesností, aby prošly klasifikací a byly vyznačeny. Přesnost dopravního značení `Stůj, dej přednost v jízdě` se pohybovala kolem 74%.

Při testování jsem si povšimnul, že dopravní značení dej přednost v jízdě mělo téměř bezchybnou klasifikaci, pouze v jednom ze 150 případů nebyl model rozpoznán s přesností větší 85%, tedy v jednom z případů byl model vyhodnocen jako falešně negativní. Pro značení hlavní cesty dosahoval

model také velmi přesného rozpoznávání 95%, jenže oproti značce dej přednost se zde vyskytovaly chybně určené objekty. Tyto chybně detekované objekty nicméně představovaly pouhé 0,5% z celkového počtu všech značení hlavní cesty. Pro křižovatku s vedlejší pozemní komunikací byla přesnost rozpoznání téměř 90%.

6.5 Rozpoznávání dodatkových tabulí

Námi natrénovaný model dokáže rozpoznat značky na křižovatkách s přesností 88,7 %, nicméně stejně důležitou značkou, jako značka samotná, je dodatková tabule, která určuje tvar křižovatky. V některých případech totiž nemusí značka hlavní cesty znamenat, že máte absolutní přednost. Představme si, že křižovatka se stáčí doprava, v takovémto případě sice máme přednost, pokud bychom ale chtěli jet rovně, tehdy platí pravidlo pravé ruky. Automobil přijíždějící zprava má přednost vzhledem k nám. Pro zdokonalení modelu byly hledány tyto dodatkové tabule a informace z nich porovnány.

Předem nalezená pozice svislé dopravní značky byla využita pro zmenšení oblasti, ve které je dodatková tabule vyhledávaná. Tvar křižovatky je rozpoznáván pouze pod detekovanou značkou, nikoli v celém obraze.

Na takto nalezené oblasti, ve kterých by se mohly dodatkové tabulky nacházet, byly použity metody knihovny OpenCV, dříve zmíněné v kapitole 4.4. Nejprve byl obraz mírně rozostřen, čímž bylo docíleno snažší vyhledávání v obraze. Následně byla vytvořena maska, viz. obrázek 6.3a, jež vyhledávala v oblasti tvary, které měly barvu od černé až po tmavě šedou. Dolní hranice prahu byla pevně daná, tj. černá barva, pro ideální nalezení horní hranice bylo třeba testů. Pro určení barev byl využit barevný model HSV, který nejvíce odpovídá lidskému vnímání barev. Skládá se ze tří částí - Hue, Saturation, Value. První z nich popisuje odstín, druhá část popisuje sytost barvy a poslední je hodnota jasu. Každá z těchto částí může mít hodnoty od 0 do 255 bitů. V tabulce 6.2 můžeme vidět výsledky experimentů, kdy byla převážně měněna horní hranice barevného spektra.

Tabulka 6.2: Výsledky experimentů změny horní hranice prahu

Odstín (Hue)	Sytost (Saturation)	Jas (Value)	Úspěšnost rozpoznání
170	55	100	31,8%
125	40	255	22,7%
70	60	100	40,9%
180	30	65	59,1%
255	55	60	81,8%

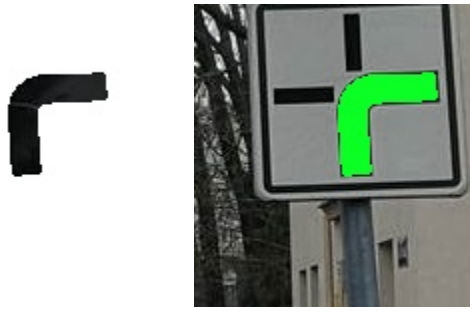
Po nalezení nejúspěšnější horní hranice prahu byla vytvořena maska, jež přetváří rozpoznanou oblast na černobílý obraz, viz. obrázek 6.3a. Pixely s hodnotami mezi dolní a horní hranicí prahu byly označeny v masce bílou barvou, ostatní černou barvou. Poté byly použity morfologické ope-

race, které bývají aplikovány na binární obrazy. Na vstupu je zadána maska, tvořící původní obraz a stavební prvek (čtverec, elipsa nebo kříž), který byl v obraze hledán. V našem případě se jedná o elipsu. Pro získání požadovaného tvaru z obrazu byla na vstupní masku aplikována binární transformace otevření a uzavření, výsledkem obou operací je zjednodušený obraz s méně detaily. Otevření vyfiltruje z obrazu malé objekty, uzavření naopak objekty blízko sebe spojí a zaplní tak malé díry. Ve výstupním skriptu se využívá každé z těchto transformací dvakrát pro každou prohledávanou oblast pod nalezenou značkou, díky nim jsou nalezeny souřadnice vyhledávaného tvaru, které jsou následně využity pro určení tvaru křižovatky.

(a) Maska (b) Vstupní obraz Morfologické operace (c) Výstupní obraz Morfologické operace

Obrázek 6.3: Ukázka přetransformovaného vstupního obrazu

Pro vykreslení tvaru byla užita metoda na získání obrysu. Tvar nalezeného objektu byl rozpoznán a do původního obrazu byl zapsán tvar křižovatky, viz. obrázky 6.4. V případě, že nebyl nalezen žádný objekt, dodatková tabulka pod značkou neexistuje. Pokud byl nalezen objekt, který by pod značkou neměl existovat, je následně použito dodatečné rozpoznání oblasti skrze metodu srovnání se vzorem SSIM.



(a) Nalezený objekt (b) Objekt vyznačený v původním obraze

Obrázek 6.4: Ukázka nalezení a rozpoznání tvaru křižovatky

Pro porovnání dříve nerozpoznané oblasti bylo vytvořeno 6 nových datových sad, v nichž jsou rozděleny obrázky dle tvaru křižovatky. Ukázka obsahu těchto sad je na obrázku 6.5. Původně byly porovnávány pouze tvary křižovatek uložené v jedné malé datové sadě, nicméně úspěšnost

rozpoznání nevyhovovala, jelikož byly často chybně rozpoznány tvary i na místech, kde se dodatkové tabule vůbec nevyskotovaly.

(a) Bez dodatkové tabule (b) Hlavní cesta vpravo (c) Hlavní cesta zprava (d) Hlavní cesta zleva

Obrázek 6.5: Ukázka obsahu datových sad pro dodatkové tabule

V této práci byl oblasti změněn výstupní tvar, který byl následně porovnán za pomoci klasifikátoru obrazu SSIM. Structural Similarity Index (`SSIM`)[17] je jednotka, která vyjadřuje podobnost dvou obrazů na základě podobného rozložení či hustoty pixelů. Tento index nabývá hodnot -1 až 1, kde 1 vyjadřuje totožné obrazy. Pro porovnání jsou zapotřebí dva obrázky stejného tvaru a velikosti. Pokud bychom měli obrazy jiných velikostí, je zapotřebí jednomu z nich upravit velikost.

Porovnání probíhalo procházením 3 datových sad rozdělených mezi značky určující přednost a značky říkající, že přednost musíte dát jiným. Pokud byla oblast otestována na všechny druhy tvarů křižovatky daných typem cesty, následovalo uložení vzhledově nejpodobnějšího tvaru a vepsání tohoto tvaru křižovatky do původního obrazu do levého horního rohu.

Obrázek 6.6: Ukázka rozpoznání tvaru křižovatky

Na obrázku 6.6 je vidět směr hlavní cesty v levém horním rohu popsaného jako ”vpravo” pro křižovatku označenou značkou hlavní pozemní komunikace směřující doprava. Pro hlavní cestu byly

dálo použito označení ”rovně” a ”doleva”. Pro vedlejší cesty pak byly stavy ”zprava” pro cestu vedoucí zprava rovně, ”zleva” a ”zleva vpravo”.

(a) (b)

Obrázek 6.7: Ukázka chybného rozpoznání tvaru kižovatky

Na obrázku 6.7a byl tvar křižovatky chybně vyhodnocen, nejprve nebyl tvar správně nalezen a poté chybně vyhodnocen porovnáním obrazu. Tvar hlavní cesty zprava a zleva mají velmi podobnou strukturu. Klasifikátor obrazu rozhodl, že více se tabulce podobá cesta zprava, nikoli zleva. Výsledky obou porovnání se moho lišit v tisícinách, desetitisícinách procent, bohužel byla tabulka rozpoznána chybně. Pro obrázek 6.7b platí to, že byl nalezený tvar křižovatky vedlejší cesty vyhodnocen jako ”zleva”.

Jak již bylo zmíněno, detekce dodatkových tabulek přímo závisí na nalezení samotné značky. V případě, že značka nalezena nebyla, samotná tabulka není vyhledávána. Přesnost rozpoznání tvaru křižovatky se pohybuje okolo 92%. Původně nebylo rozpoznání značek a pouhé porovnávání tabulek s obrázkovými sadami příliš úspěšné. Důvodem mohla být malá datová sada a velký nedostatek oblastí bez tabulky, jež způsobil chybné vyhodnocení tvaru. V mnoha případech se stávalo, že na místě, kde se žádná dodatková tabule nenacházela, byl tvar chybně rozpoznáván. Z tohoto důvodu byla předchozí metoda využita pouze v případě, kdy program nemůže s jistotou určit tvar.

Tabulka 6.3: Výsledky testování modelu s vyhodnocením dodatkových tabulí

Název jednotky	Hodnota
Pravdivě pozitivní	91,4%
Falešně pozitivní	3,5%
Falešně negativní	5,1%
Precision	0,961
Recall	0,906
F1 Score	0,933

6.6 Zhodnocení výsledků

Model vytvořený pro rozpoznávání čtyř druhů dopravního značení má pravdivě pozitivní výsledky v 89% případů. Toto číslo bylo ovlivněno v prvé řadě klasifikací dat nad hodnotou 0,85, klasifikátor označil pouze objekt, které byly rozpoznány s minimální přesností 85%, což by vysvětlilo, vysoký počet případů nenalezení značky `stůj, dej přednost v jízdě`, která byla v datové sadě obsažená minimálně. Vezmeme-li v úvahu, že objekty rozeznané alespoň s přesností 85% a více tvořilo 91% celkových případů, můžeme prohlásit tento detektor za velice úspěšný. Celková přesnost rozpoznávaní udávaná jednotkou F1 zobrazená v tabulce 6.1 je v tomto případě 94%.

Druhá část práce zaměřená na rozpoznání tvaru křižovatky nicméně zprvu takto úspěšná nebyla. Nedostatečně zastoupená porovnávací datová sada tvořená obrázky nedostatečné kvality vykazovala úspěšnost rozpoznání necelých 60%, často se vyskytovaly chybně rozeznané tvary, převážně na místech, kde se vůbec tabule nevyskytovala. Od tohoto způsobu bylo upuštěno a vytvořen časově náročnější model, který využívá pro rozpoznání tvaru křižovatky masku aplikovanou na obraz. Pokud nebyl nalezen, je pro klasifikaci tvaru využito 6 nezávislých datových sad. První skupinu tvoří 3 datová sada pro hlavní cestu a druhou vedlejší cesta také o 3 datových sadách pro každý směr křižovatky.

Pokud by se stalo, že se v procházené oblasti pod značkou dodatková tabulka vůbec nenacházela, ale byl by zde nalezen objekt, nedá se s jistotou říct, že bude správně vyhodnocena. Těchto případů by mělo být minimum. Na obrázku 6.8 můžeme vidět speciální tvar křižovatky. Jedná se v takovém případě o cestu doprava či rovně? Program v tomto případě rozhodl, že se jedná o hlavní cestu, která vede doprava.

Obrázek 6.8: Ukázka možného tvaru křižovatky neobsaženého v datových sadách

Celková úspěšnost detektoru při zahrnutí rozpoznání tvaru křižovatky klesla na 93,3%, viz. F1 Score z tabulky 6.3. Rychlost detekce modelu se pohybuje okolo 0,2 vteřiny na snímek, jelikož bylo užito algoritmu Faster R-CNN, nicméně čas může být delší kvůli procházení dalších přibližně 2000 snímků ze 7 nových datových sad.

Kapitola 7

Závěr

Výsledkem této práce bylo navrhnout detektor a klasifikátor dopravního značení v obrazech a následně model řádně otestovat a zhodnotit výsledky.

V této práci byla zjednodušeně popsána funkčnost neuronových sítí, jejich stavba a princip učení. Dále se práce zaměřila na konvoluční neuronové sítě, které byly zvoleny jako řešení pro náš problém, detekci značek na křižovatkách. Byl popsán princip, vrstvy a algoritmy detekce CNN.

Na počátku práce byla datová sada pro vytvoření modelu, která pro čtyři vyhledávané typy dopravního značení sčítá 2153 snímků. Rovněž byla vytvořena nezávislá testovací datová sada pro výsledné testování přesnosti detekčních sítí.

Pro vytvoření programu, resp. modelu byl zvolen framework TensorFlow s využitím jeho rozhraní pro detekci objektů s nímž práce byla velmi příjemná. Použil jsem algoritmus konvoluční neuronové sítě známý jako Faster R-CNN, jenž byl trénován na 200 tisíc kroků. Trénování modelu se nicméně nepohybovalo v řádech hodin, jak by na dvou grafických kartách NVIDIA GeForce RTX 2080 rozhodně mělo, nýbrž v řádech dnů, přesněji 5 dnů a 13 hodin. V průběhu trénování bylo zjištěno, že při použití Tensorflow 1.5 verze i přes předchozí nainstalování GPU verze, se trénování provádí pouze na procesoru.

Chybovost rozpoznání jednotlivých značek byla téměř nulová. Značka dej přednost v jízdě, měla největší zastoupení v trénovací datové sadě a dosahovala takřka 100% úspěšnosti. Podobně na tom byla značka hlavní pozemní komunikace. Problém nicméně nastal při zjišťování dat z doplňkové tabulky, jelikož tvar křižovatky byl samozřejmě rozdílný. Nalezení správného postupu pro zjištění tvaru křižovatky bylo nejsložitější částí celé práce, jelikož rozpoznání záviselo převážně na kvalitě kamery a světelných podmínkách. Úspěšnost rozpoznání tvaru křižovatky se původně pohybovala okolo 60%. Tato hodnota byla zapříčiněna malou datovou sadou o obrazech pochybné kvality, jež často nebyly schopny správně rozpoznat tvar. Z toho důvodu bylo vytvořeno rozpoznání na základě nalezeného tvaru křižovatky. Ve chvíli, kdy nalezený objekt neodpovídal žádnému hledanému tvaru bylo použito 6 nezávislých datových sad pro všechny tvary křižovatek. Po nasazení nové verze do

programu se rozpoznání tvaru křižovatky pohybuje kolem 92%. Po rozpoznání tvaru křižovatky je do levého horního rohu zapsán tvar, resp. směr křižovatky.

Pro detektor byly vytvořen jeden skript, který umožňuje vybrání typu vstupních dat. První možností je rozpoznání dat v obrázkové testovací sadě, druhá ze vstupního videa vytváří zcela nové výstupní s detekovanými a klasifikovanými značkami a poslední poskytuje rozpoznání dat v reálném čase.

Literatura

1. LIN, Huei-Yung; HUANG, Shu-Chun. Traffic Sign Detection and Recognition for Driving Assistance System. *Advances in Image and Video Processing*. 2018-06, roč. 6. Dostupné z DOI: 10.14738/aivp.63.4603.
2. MILOŠ ULDRICH, Tomáš Jurczyk. *Neuronové sítě a jejich využití* [online] [cit. 2020-04-21]. Dostupné z: https://www.systemonline.cz/clanky/neuronove-site-a-jejich-vyuziti-1.htm.
3. VOLNÁ, E. *Neuronové sítě 1*. 2008. Ostrava: Ostravská univerzita v Ostravě. Vydání: druhé.
4. *7 Types of Neural Network Activation Functions: How to Choose?* [Online] [cit. 2020-04-21]. Dostupné z: https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right.
5. SHARMA, Sagar. *Activation Functions in Neural Networks* [online] [cit. 2020-04-21]. Dostupné z: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.
6. SHARMA, Himanshu. *Activation Functions : Sigmoid, ReLU, Leaky ReLU and Softmax basics for Neural Networks and Deep Learning* [online] [cit. 2020-04-21]. Dostupné z: https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e.
7. HAO, Shijie; ZHOU, Yuan; GUO, Yanrong. A Brief Survey on Semantic Segmentation with Deep Learning. *Neurocomputing*. 2020, roč. 406, s. 302–321. ISSN 0925-2312. Dostupné z DOI: https://doi.org/10.1016/j.neucom.2019.11.118.
8. POKHARNA, Harsh. *The best explanation of Convolutional Neural Networks on the Internet!* [Online] [cit. 2020-04-21]. Dostupné z: https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8.
9. GANDHI, Rohith. *R-CNN, Fast R-CNN, Faster R-CNN, YOLO - Object Detection Algorithms* [online] [cit. 2020-04-23]. Dostupné z: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

10. ALAMSYAH, Derry; FACHRURROZI, Muhammad. Faster R-CNN with Inception V2 for Fingertip Detection in Homogenous Background Image. *Journal of Physics: Conference Series*. 2019-03, roč. 1196, s. 012017. Dostupné z DOI: 10.1088/1742-6596/1196/1/012017.

11. WILLEMS, Karlijn. *TensorFlow Tutorial For Beginners* [online] [cit. 2020-01-20]. Dostupné z: https://www.datacamp.com/community/tutorials/tensorflow-tutorial.

12. *Tensorflow* [online] [cit. 2020-01-19]. Dostupné z: https://github.com/tensorflow/tensorflow/README.md.

13. *TensorFlow Object Detection API* [online] [cit. 2020-01-19]. Dostupné z: https://github.com/tensorflow/models/tree/master/research/object_detection/README.md.

14. *TensorFlow Detection Model Zoo* [online] [cit. 2020-01-19]. Dostupné z: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.

15. *LabelImg* [online] [cit. 2020-01-19]. Dostupné z: https://github.com/tzutalin/labelImg.

16. *OpenCV* [online] [cit. 2020-01-20]. Dostupné z: https://opencv.org.

17. *SSIM: Structural Similarity Index* [online] [cit. 2020-04-23]. Dostupné z: https://www.imatest.com/docs/ssim/.

Příloha A

Obsah archivu

- Datové sady
  - images/train
  - images/test
  - intersection
    - Left
    - Right
    - Straight
    - fromLeft
    - fromRight
    - leftToRight
- Natrénovaný model
  - inference_graph/frozen_inference_graph
- Značkovací mapa
  - training/labelmap.pbtxt
- Skripty
  - classes
    - IntersectionShapeRecognition.py
    - SignDetection.py
  - generate_tfrecord.py
  - Object_detection.py
  - train.py